Project Proposal

- One proposal per team
- Presentation (4 or 5 slides)
  - 10 min presentation + 2 min questions
- Goals
  - What exactly do you want to do?
  - Why what you want to do is important?
- Challenges
  - What are the challenging parts in your project?
- Description of project.
  - How do you plan to do it? (preliminary ideas)

- 5% of your final score
- Due: Midnight, 09/16 (Wednesday)
- Grading based on
  - How clear you are about your project (what to do)?
  - Background studies (related work)

Project Midterm Presentation

- Slides due midnight 10/19 (Monday)
- Hard deadline
- Midterm presentation (10% in final grade)
  - Short summary of your project idea
  - What have you done so far? progress
  - What are the results you have? What problems do you have?
  - What is your next step before final?
  - How to demonstrate your project at the final presentation?
  - Who is doing what? How do you divide the work?
- Grading based on progress
  - Detailed algorithms/design, preliminary results, etc

ECE 692
Power-Aware Computer Systems

End-to-End CPU Utilization Control

Prof. Xiaorui Wang

Problems in Critiques

- You should try to find concrete problems / holes / wrong assumptions in the paper
- Two kinds of critiques are not interesting
  - Too minor: formatting, spelling errors
    - E.g., should add a space to somewhere in the paper
  - Too general and meaningless
    - E.g., I don’t understand this paper (because you didn’t read it carefully?)
    - Those are bad reviews

Recap of Last Class

- Feedback control
  - What is feedback control?
  - Why feedback control?
- Design of a utilization controller
  - System modeling
  - Performance specs/metrics
  - Controller design
- Summary
Distributed Real-Time Systems

- An (end-to-end) task is composed of multiple subtasks running on multiple processors
  - Message/event
  - Remote method invocation
- Subtasks are subject to precedence constraints
  - Task = a chain/tree/graph of subtasks
  - E.g. ship navigation

Motivation of Utilization Control

- CPU utilization: a trade-off
  - Too high → system overload → crash
  - OS frozen by higher-priority real-time threads
  - Too low → poor application QoS
- Schedulable utilization bound
  - Utilization ≤ bound → meet all deadlines
  - Highest possible utilization with deadline guarantee
  - Uncertainties:
    - Unpredictable exec times (e.g. influenced by sensor data)
    - External resource contention (e.g. Denial of Service attacks)
  → Must maintain desired utilization under uncertainty!

The Common End-to-End Task Model in Distributed Real-Time Systems

- Periodic task T_i = a chain of subtasks {T_ij} on different processors
  - All subtasks run at a same rate
  - End-to-end deadline is divided into a set of sub-deadlines
  - Task rate can be adjusted within a range
  - E.g., online video can trade-off between quality and rate
  - Higher rate → better video quality & higher CPU utilization

End-to-End Utilization Control

- CPU utilization: a trade-off
  - Too high → system overload → possible crash
  - Too low → poor performance (e.g. poor video quality)
  - Utilization ≤ schedulable bound → meet all sub-deadlines
- End-to-end utilization control
  - Utilizations of all proc ≤ bounds → meet all end-to-end deadlines
  - Constraints on task rates
  - Stability assurance

Challenges of End-to-End Utilization Control

- Multi-Input-Multi-Output (MIMO) control
- Utilizations are coupled due to end-to-end tasks
  - Rate change affects all processors in the task chain
  - Constraints on task rates
  - Stability assurance

EUCON – Centralized Control Algorithm

- EUCON (End-to-end Utilization CONtrol)
  - Designed based on Model Predictive Control (MPC) theory
  - Invoked periodically to control the utilizations of all processors

Controller

Desired utilization bounds

Controlled variables: CPU utilizations

Manipulated variables: task rates

Model Predictive Controller

Desired rate range for tasks (constraints)

Allowed rate range for tasks (constraints)
### Design Steps of Model Predictive Controller

1. Derive the dynamic model of the controlled system
2. Design the controller
   - **Design objective:**
     \[
     \min \sum_{i=1}^{m} (B_i - u_i(k))^2
     \]
   - subject to rate constraints: \(R_{\text{min}}(k) \leq R(k) \leq R_{\text{max}}(1 \leq j \leq m)
3. Analyze system stability

### Step 1: Dynamic System Model

- **New utilization = old utilization + change**
- **Utilization change = actual exec time x task rate change**

\[
\begin{align*}
    u_i(k) &= u_i(k-1) + g_i \Delta r_i(k-1) \\
    u_j(k) &= u_j(k-1) + g_j \Delta r_j(k-1)
\end{align*}
\]

- \(g_i\): estimated execution time of \(T_i\)
- \(g_j\): actual execution time / estimation
- Models uncertainty in execution times

### Step 2: Controller Design

- **Core of MPC is a constrained optimization solver**
- **Difference with reference trajectory**

\[
\begin{align*}
    \Delta u_i(k+1) &= \text{Optimization Result} \\
    \Delta u_j(k+1) &= \text{Optimization Result}
\end{align*}
\]

### Step 3: Analytical Stability Assurance

- **Stability:** converge to desired bounds from any initial condition
- **Uncertainty gain:** \(g = \) actual execution time / estimation
- **Stability condition:** tolerable range of uncertainties
  - E.g., \(0 < g < 5.95\)

### Effect of System Gain

- **System is designed based on estimated exec times**
- **In run-time, real exec times are different from estimations**
- **The difference is called system gain**

### Utilization Modeling on One Processor

- **CPU utilization is the sum of the utilizations of all tasks**
- **Real task execution times differ from their estimations in a ratio \(g\):**
  - Running on a different processor
  - Influenced by external inputs
- **In the \(k\)th time period**
  \[
  u(k) = u(k-1) + g \Delta u(k-1)
  \]
  \[
  = u(k-1) + g_r \Delta r(k-1) + e_r \Delta r(k-1) - e_r \Delta r(k-1)
  \]

### CPU Utilization

- **Stability experiments of an example system**
- **Stability range**
  - Real exec time
  - System unable
  - System usable
  - System usable
Experiments: Workload Uncertainties
- 12 tasks (25 subtasks) running on 4 Pentium IV processors (Linux)

Control Problem Decomposition
- Problem decomposition for processor \( P_1 \)
  - Direct neighbors: \( P_2 \)
  - Concerned tasks: \( T_1 \) and \( T_2 \)
  - Indirect neighbors: \( P_3 \)
  - Neighbors which start \( P_1 \)'s concerned tasks
- Neighborhood of \( P_1 \)
  - Includes itself, direct neighbors and concerned tasks
  - Neighborhoods can have overlap

Dynamic System Model
- New utilization = old utilization + change
- Utilization change = actual exec time x task rate change
- System model:
  \[
  u_t(k) = u_t(k-1) + g_t c_t \Delta r_t(k-1) \\
  u_s(k) = u_s(k-1) + g_s (c_s \Delta r_s(k-1) + c_{s2} \Delta r_{s2}(k-1))
  \]
- \( c_t \): estimated execution time of \( T_t \)
- \( g_t \): actual execution time / estimation
- Models uncertainty in execution times

Decentralized End-to-End Utilization Control
- Limitations of centralized control in large system
  - Poor scalability
  - Communication overhead
  - Intolerable communication delay
  - Single point of failure
- DEUCON
  - Distributed Model Predictive Control (DMPC) theory
  - Localized control decision
  - Neighborhood coordination

Peer-to-Peer Neighborhood Control
- Localized feedback control loop
  - Local MPC control computation
  - Only considers processor \( P_1 \) and task \( T_1 \)
  - Data exchange with neighbors
  - Exchanges rate with indirect neighbor \( P_3 \)
  - Unlimited system size vs. limited neighborhood size

Controller Design
- Least Squares Solver
- Model Predictive Controller
- Core of MPC is a constrained optimization solver
- Desired trajectory for \( u(k) \) to converge to \( B \)
- Cost function:
  \[
  J(k) = \sum_{i=1}^{N} \left( \sum_{r=1}^{R} \Delta u_r(k+i) - \Delta u_r(k+i+1) \right)^2 + \sum_{i=1}^{N} \left( \sum_{r=1}^{R} \Delta u_r(k+i) - \Delta u_r(k+i+1) \right)^2
  \]
Global System Stability

- Stability
  - Utilizations converge to schedulable utilization bound whenever feasible
  - Guaranteed by Distributed Model Predictive Control (DMPC) theory
- Stability condition
  - How far can real exec times deviate from estimations?
  - Within an analytical range → system is stable
  - Example system: stable when \( 0 < \text{gain} \leq 2 \)

Simulation Setup

- 10 processors
- 21 tasks
- 14 end-to-end tasks
- 40 subtasks in total
- RMS scheduling
  - Sub-deadline = period
- Release guard
  - \( \text{ietf} \) (Inverted Exec Time Factor)
    - \( \text{ietf} = \frac{1}{\text{Real exec-time}} \)
    - Evaluates the robustness of DEUCON

Overhead Comparison

- Controller execution time
  - Polynomial in problem size
- Communication overhead
  - Utilization exchange
  - Task rate exchange
- DEUCON is more scalable
  - Larger system size → Bigger difference

Scalability of DEUCON in Large Systems

- Simulation with randomly generated workloads
  - Each processor has 5 subtasks on average
- Controller computation overhead
  - Polynomial in neighborhood size and # of concerned tasks
- Communication overhead

Performance Comparison

- Performance is close
  - Minor difference only in extreme cases
  - Insignificant difference only when \( \text{ietf} = 6 \) (i.e. real exec times are 1/6 of the estimations)

Real-Time Guarantees

- 21 tasks (40 subtasks) running on 10 processors
  - Global execution time variations
  - Local execution time variations
Summary

- EUCON provides analytical utilization guarantees despite uncertain execution times
- Based on Model Predictive Control (MPC)
- Handles coupling among processors
- Enforce constraints on task rates
- DEUCON: decentralized utilization control
- Decentralized control vs. centralized control
  - Decentralized control is much more scalable
  - Performance is close